Bridging the sanitization gap:
KMSAN for user-mode Linux atop an ARM-based microkernel

Author: Pedro Guerra Lourenço **Advisor:** Sandeep Tamrakar

- **Kernel Memory Sanitizer** is designed to detect Use-of-Uninitialized-Memory (UUM) bugs.
- KMSAN is restricted to monolithic Linux kernels on x86_64 architectures.
- Our solution helps bridging this sanitization gap by porting KMSAN to a user-space Linux kernel service (LibLinux) in the ARM64-based **HarmonyOS** (HOS) microkernel ecosystem.

Introduction

- UUM bugs rarely crash systems, but are the main cause of canary and KASLR pointer leak.
- Impractical to initialize all allocated data (performance).
- KMSAN detection mechanism uses bit-precise **Shadow Memory** to track initialization states and **Origin Tracking** to pinpoint allocation sites.
- KMASN improves fuzzing routines by converting silent leaks into actionable reports.

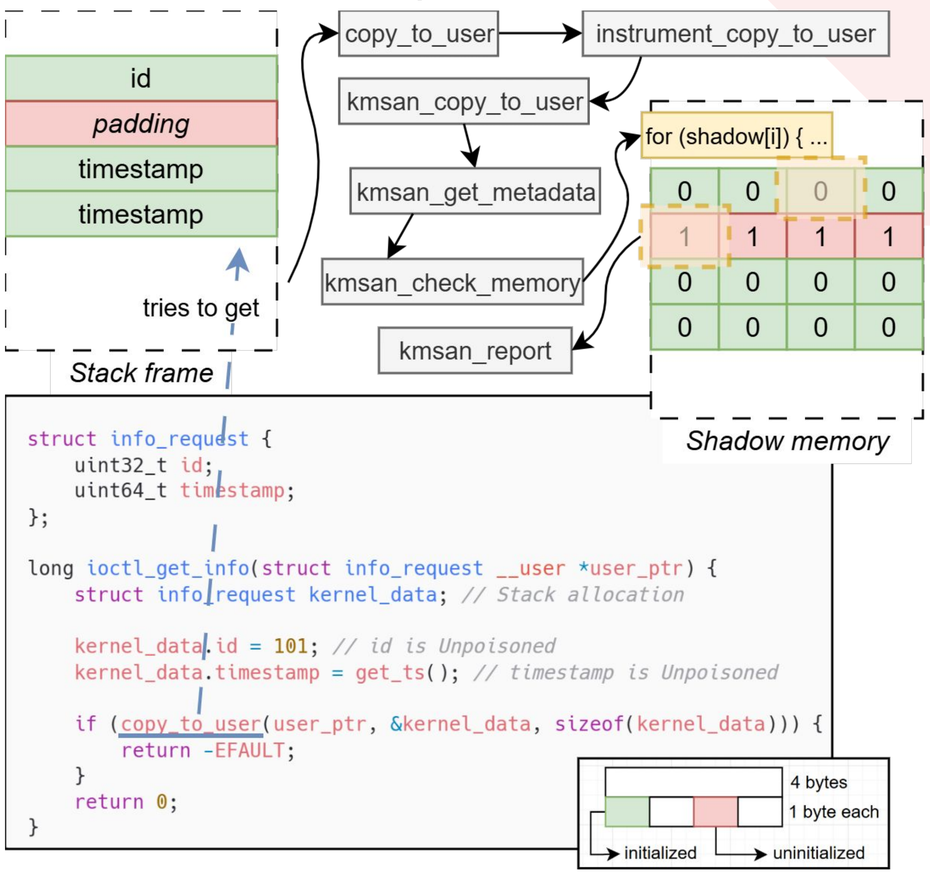

Figure 2: KMSAN check-and-report flow. Before data reaches a "sink", KMSAN its initialization state. If uninitialized bits are detected, the system triggers a report identifying the leak's origin.

The solution

- Backported KMSAN from v6.1 to v5.10.
- Applied the instrumentation pass to LibLinux.
- Re-engineered x86-specific metadata logic for the ARM64-based HOS context.
- Decoupled monolithic dependencies to operate within a user-space microkernel container.
- Custom changes in thread creation, memory reservation, TLS data access, data copy to/from IO, valid VA ranges, hardware memory mapping.

Conclusion

- Demonstrated that complex monolithic sanitizers can be abstracted for microkernel architectures.
- Reduced the sanitization gap for ARM64, providing UUM detection for HOS services.
- Established a reference for porting advanced sanitizers to emerging microkernel ecosystems.

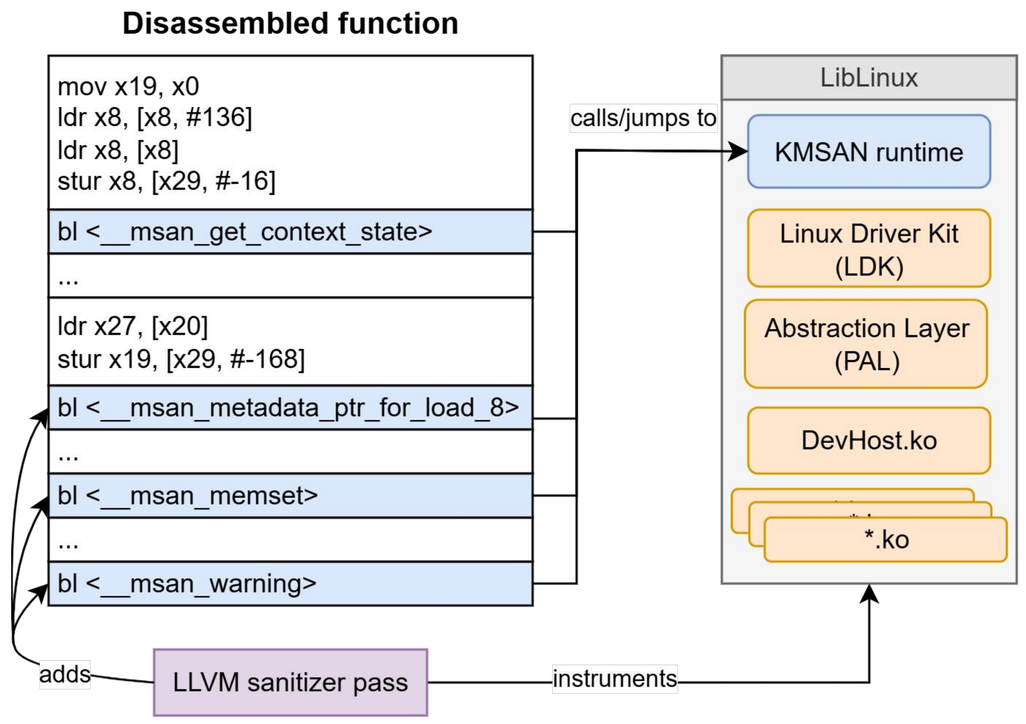

Figure 1: KMSAN-instrumented function. LLVM performs a sanitization pass over the code's IR, injecting hooks that call the KMSAN runtime. At execution, these hooks manage metadata propagation and shadow updates across all memory operations.

The problem

- KMSAN tethered to x86 and monolithic kernel.
- Security blind spot: the dominant architecture for Mobile, IoT, and Edge is ARM64.
- Weaker bug detection → less secure software
- HOS services cannot leverage KMSAN. They run atop the ARM64 HongMeng microkernel.
- Linux runs as a library. Memory management is offloaded to the microkernel.

The challenges

- LLVM instrumentation pass for LibLinux in a BitBake-based build pipeline with 2700+ tasks.
- HOS re-implemented memory allocation and mapping, scheduling, interrupts, sync. functions.
- Intricate device interactions and MMIO.
- Usage of an older Linux version: 5.10.

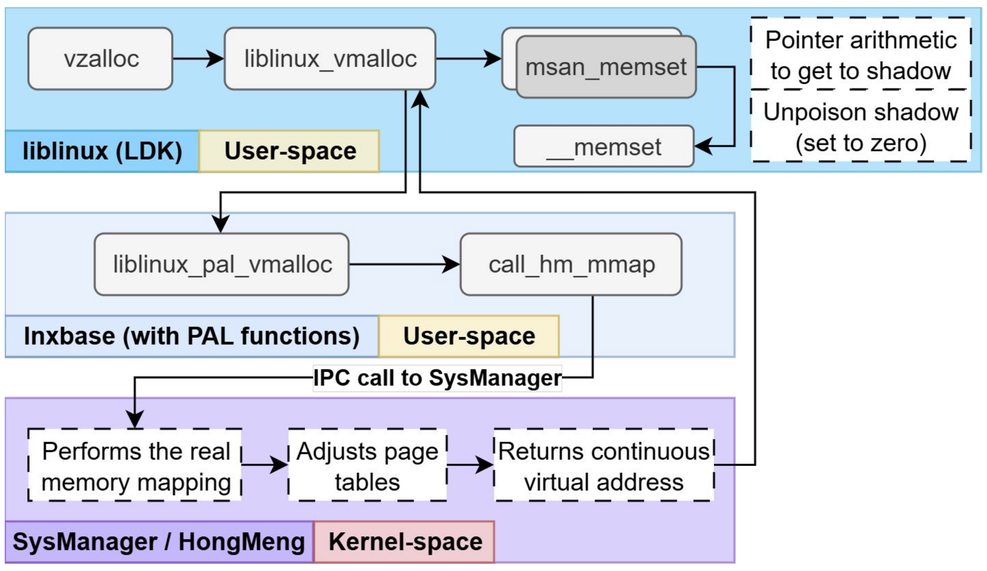

Figure 3: vzalloc and KMSAN boundaries. vzalloc call originates from LibLinux, passes through the PAL to SysManager for memory allocation and mapping. KMSAN stays in LDK.

Helsinki System Security Lab (HSSL)

HSSL drives renewal and mastery in the field of platform and device related security technologies, especially for Huawei consumer devices such as mobile phones, laptops, televisions and automotive. We do research in topics such as hardware-assisted isolation and integrity, as well as in operating system protection (hypervisor, TEE, secure enclaves and kernel hardening). We also carry expertise in cryptography and systems security functionality such as device key management (PKI), device attestation and key-store solutions.

HUAWEI